(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2000020481 A**

(43) Date of publication of application: **21.01.00**

(51) Int. Cl **G06F 15/16**
G06T 1/20

(21) Application number: **10190096**

(22) Date of filing: **06.07.98**

(71) Applicant: **HITACHI LTD**

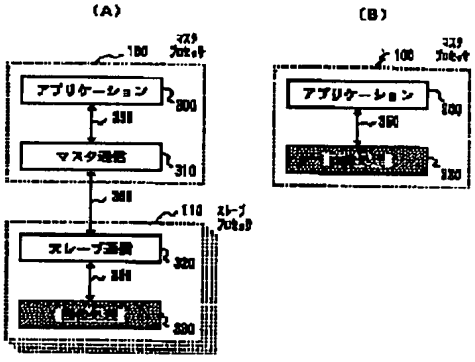
(72) Inventor: **YOSHIDA MASASHI**
IKEDA KOJI
TAKANE ATSUSHI
SATO NORIO

(54) PARALLEL IMAGE PROCESSING SYSTEM

(57) Abstract:

PROBLEM TO BE SOLVED: To attain the miniaturization/cost-down of a device by providing the device through minimum processes without performing recompiling or the like so as to execute a program, which is operated between parallel processors, through a single processor.

SOLUTION: An interface for calling a master communication program from an application 300 is made equal with an interface for calling an image processing program from a slave communication program, the image processing program is directly called from the application 300 and the image processing program can be executed on a master.



COPYRIGHT: (C)2000,JPO

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2000－20481
（P2000－20481A）
(43)公開日　平成12年１月21日(2000.1.21)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'（参考）
Ｇ０６Ｆ　15/16	３８０	Ｇ０６Ｆ　15/16	３８０Ｚ	５Ｂ０４５
Ｇ０６Ｔ　1/20		15/66	Ｋ	５Ｂ０５７

審査請求　未請求　請求項の数１　ＯＬ　（全９頁）

(21)出願番号　特願平10－190096

(22)出願日　平成10年７月６日(1998.7.6)

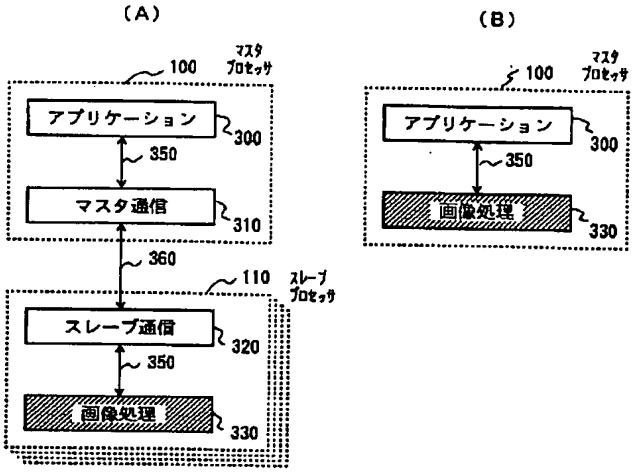
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　吉田　昌司
茨城県日立市大みか町七丁目１番１号　株式会社日立製作所日立研究所内
(72)発明者　池田　光二
茨城県日立市大みか町七丁目１番１号　株式会社日立製作所日立研究所内
(74)代理人　100068504
弁理士　小川　勝男

最終頁に続く

(54)【発明の名称】　並列画像処理システム

(57)【要約】
【課題】並列プロセッサで動作するプログラムを、単一プロセッサで実行するように、再コンパイルなどを行うことなく、最小の工数で実現し、装置の小型化／装置コストの削減を図る。
【解決手段】アプリケーションからマスタ通信プログラムを呼び出すインタフェースと、スレーブ通信プログラムから画像処理プログラムを呼び出すインタフェースを同一にし、アプリケーションから画像処理プログラムを直接呼び出して、画像処理プログラムをマスタ上で実行することを可能とする。

図　3



1

【特許請求の範囲】

【請求項１】マスタプロセッサと、

前記マスタプロセッサと同一の命令で動く複数のスレーブプロセッサとからなる並列画像処理システムにおいて、

前記マスタプロセッサ上で動作する処理部と、

前記マスタプロセッサ上で動作し、画像をそれぞれの前記スレーブプロセッサ用に分割し、それぞれの前記スレーブプロセッサとの間でデータ転送を行うマスタ通信部と、

前記スレーブプロセッサ上で動作し、前記マスタプロセッサとのデータ転送を行うスレーブ通信部と、

前記スレーブプロセッサ上で動作し、分割された画像に対して画像処理をする画像処理部とを有し、

前記処理部と前記マスタ通信部間のデータ転送手段が、前記スレーブ通信部と前記画像処理部間のデータ転送手段と同一である並列画像処理システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、データ処理計算機の技術分野に属し、マスタ／スレーブ型並列プロセッサ画像処理装置上のプログラムの構成に関している。

【０００２】

【従来の技術】複数のプロセッサを用いて処理を行う並列プロセッサは、性能向上を図る技術として、かなり以前より研究／開発されてきた。また、画像処理は画素毎に独立な処理となりやすく、処理の並列性が高い性質があるため、他の分野と比較して並列プロセッサにより大きく性能を向上でき、並列プロセッサを用いた画像処理装置も数多く研究／開発されてきている。

【０００３】従来の並列プロセッサ技術は、プログラムを作る工数を減らし、またなるべく多くの分野で用いられるような汎用性を持たせるために、単一プロセッサ用のプログラムを並列プロセッサ用のプログラムに自動変換する、並列コンパイラという技術がある。並列コンパイラに関した文献／特許は、“佐藤誠他、「パターン情報処理用マルチマイクロプロセッサシステムＰＸ－１」、信学論Vol.64,No.11, p.1021－1028 (Nov.,1981)”, “Robert G. Babb、「並列処理マシン用FortranのＨＰＦ、業界標準の地位を目指す」、日経エレクトロニクス、no. 581 (1993.5.24)”、特開平9－91258号, 特開平8－328871号, 特開平7－311746号, 特開平6－26481号などがある。

【０００４】また、並列プロセッサを実用化するためには、データを各プロセッサにどのように振り分けるかということが重要であり、このデータ分割の方法に関した特許としては、特開平10－69469号, 特開平8－278949号, 特開平6－325162 号などが挙げられる。

【０００５】さらに、その他並列プロセッサの制御や装置に関した文献／特許としては、“小西弘一、「並列機

2

用インターフェースＭＰＩ、アプリ開発支援機能を盛り込む、メッセージ通信の業界標準に」日経エレクトロニクス、no.605(1994.4.11)”、特開平8－249018号, 特開平8－249022号, 特開平8－249018号, 特開平8－161271号, 特開平8－16754号, 特開平7－51594号, 特開平5－257934号, 特開平5－233569号, 米国特許5050070、その他数々の文献／特許がある。

【０００６】

【発明が解決しようとする課題】最近の汎用プロセッサの性能向上はめざましく、並列プロセッサでなければ実現できなかった性能が、短期間のうちに単一プロセッサで達成される状況にある。従って、並列プロセッサを簡単に単一プロセッサへ変更できれば、装置の小型化がはかれ、コストも安くてすむ。従って、並列プロセッサとして開発したプログラムを単一プロセッサで実行する形式へ簡単に変更できる方法が課題となる。並列コンパイラという従来技術は、並列プロセッサとして開発したプログラムを単一プロセッサで実行する形式にコンパイルし直せばよいので、並列プロセッサとして開発したプログラムを単一プロセッサで実行する形式に変更することが可能である。しかし、プログラムを作る工数は減るが、並列コンパイラを作るための工数がかかり、なるべく多くの分野で用いられるような汎用性をねらわずに画像処理のみに適用するのであれば、むしろ並列コンパイラを作るための工数が大きいことが問題となる。

【０００７】また、並列コンパイラ以外で上に挙げた従来技術においては、並列プロセッサの並列性を高め性能を向上させるという観点の技術しかなく、並列プロセッサとして開発したプログラムを単一プロセッサで実行させるという観点の技術が述べられた例はない。

【０００８】本発明の課題は、並列プロセッサで動作するプログラムを単一プロセッサでも実行できるような仕組みを、再コンパイルなどを行うことなく、最小の工数で実現することにある。

【０００９】

【課題を解決するための手段】上記課題を解決するために、本発明は、マスタプロセッサと、前記マスタプロセッサと同一の命令で動く複数のスレーブプロセッサとからなる並列画像処理システムにおいて、前記マスタプロセッサ上で動作する処理部と、前記マスタプロセッサ上で動作し、画像をそれぞれの前記スレーブプロセッサ用に分割し、それぞれの前記スレーブプロセッサとの間でデータ転送を行うマスタ通信部と、前記スレーブプロセッサ上で動作し、前記マスタプロセッサとのデータ転送を行うスレーブ通信部と、前記スレーブプロセッサ上で動作し、分割された画像に対して画像処理をする画像処理部とを有し、前記処理部と前記マスタ通信部間のデータ転送手段が、前記スレーブ通信部と前記画像処理部間のデータ転送手段と同一とした。

【００１０】

【発明の実施の形態】図１は、マスタ／スレーブ型並列プロセッサのハードウェア構成例である。

【００１１】マスタプロセッサ１００，グローバルメモリ１０１，マスタバス１０２，スレーブプロセッサ１１０，ローカルメモリ１１１，スレーブバス１１２，バススイッチ１１３，スレーブプロセッサ１２０，ローカルメモリ１２１，スレーブバス１２２，バススイッチ１２３，スレーブプロセッサ１３０，ローカルメモリ131，スレーブバス１３２，バススイッチ１３３，スレーブプロセッサ１４０，ローカルメモリ１４１，スレーブバス１４２，バススイッチ１４３から成る。グローバルメモリ１０１は、１画像全体の画像データを保持しており、マスタプロセッサ１００は、その画像データをスレーブプロセッサの数である４つに分割し、ローカルメモリ１１１，１２１，１３１，１４１へ転送するが、例えばローカルメモリ１２１へ送る場合は、バススイッチ１２３をＯＮにし、それ以外のバススイッチ１１３，１３３，１４３をＯＦＦとし、マスタバス１０２，バススイッチ１２３，スレーブバス１２２を通じてローカルメモリ１２１へ転送する。このとき、例えばバススイッチ１１３はＯＦＦになっているので、スレーブプロセッサ１１０は、スレーブバス１１２を通じて、ローカルメモリ１１１から分割した画像を読みだし、画像処理し、結果をローカルメモリ１１１に書き込むことができる。マスタプロセッサ１００が他のローカルメモリ１１１，１３１，１４１へ転送する際も同様である。また、本例は、スレーブプロセッサが４つの例を示したが、数を増した場合及び数を減らした場合も全く同様に構成することができる。図２は、マスタのみの単一プロセッサのハードウェアに変更した時のハードウェア部分の構成例である。図１のマスタ／スレーブ型並列プロセッサから、スレーブ１１０，１２０，１３０，１４０及びそれに伴うローカルメモリ１１１，１２１，１３１，１４１，バススイッチ１１３，１２３，１３３，１４３を取り去った形になっている。このように、マスタ／スレーブ型並列プロセッサのハードウェアは、スレーブとその周辺回路を取り除くだけで、単一プロセッサハードウェアに容易に変更可能である。

【００１２】図３は、ソフトウェアの構成例である。（Ａ）は、図１のマスタ／スレーブ型並列プロセッサのハードウェアに対応する図、（Ｂ）は、図２の単一プロセッサのハードウェアに対応する図である。（Ａ）は、アプリケーション３００，マスタ通信３１０，スレーブ通信３２０，画像処理３３０からなる。アプリケーション３００，マスタ通信３１０は図１におけるマスタプロセッサ１００上で動作するプログラムであり、スレーブ通信３２０，画像処理３３０は、図１におけるスレーブプロセッサ、たとえばスレーブプロセッサ１１０上で動作するプログラムである。これは、残りのスレーブプロセッサ１２０，１３０，１４０でも全く同じ構成となる。アプリケーション３００はインタフェース３５０を用いてマスタ通信３１０を呼び出し、データをやり取りする。マスタ通信３１０はマスタスレーブインタフェース３６０を用いてスレーブ通信３２０を呼び出し、データをやり取りする。スレーブ通信３２０は、アプリケーション３００がマスタ通信310とデータをやり取りするインタフェースと全く同一のインタフェース３５０を用いて画像処理３３０を呼び出し、データをやり取りする。（Ｂ）はアプリケーション３００，画像処理３３０からなり、両者は図２のマスタプロセッサ１００上で動作するプログラムである。アプリケーション３００はインタフェース３５０を用いて画像処理３３０を呼び出し、データをやり取りする。このように、(A)において、アプリケーション３００からマスタ通信３１０を呼び出すインタフェース３５０と、スレーブ通信３２０から画像処理３３０を呼び出すインタフェース３５０を同一にしておくことにより、（Ｂ）のようにアプリケーション３００から画像処理３３０を直接呼び出すことができ、マスタスレーブ型マルチプロセッサの装置からスレーブを取り去ってマスタのみの単一プロセッサの装置に変更することが簡単に可能となる。

【００１３】図４は、呼び出し関係を説明する図である。(A)は、アプリケーション３００からインタフェース３５０を用いてマスタ通信プログラム３１０を呼び出し、データをやり取りする例である。（Ｂ）は、スレーブ通信３２０からインタフェース３５０を用いて画像処理３３０を呼び出し、データをやり取りする例である。アプリケーション３００にはマスタ通信プログラム３１０を呼び出す関数が書いてあり、引数データは一つの構造体のインタフェース３５０にまとめ、そのアドレスを渡す。引数データを渡されたマスタ通信プログラム３１０は、処理を行い、結果を同じインタフェース３５０にまとめて返す。スレーブ通信３２０は、同じように画像処理３３０を呼び出す関数が書いてあり、引数データは同じ形の一つの構造体のインタフェース３５０にまとめ、そのアドレスを渡す。引数データを渡された画像処理３３０は、処理を行い、結果を同じように同じインタフェース３５０にまとめて返す。

【００１４】図５は、インタフェース３５０の構成例である。(A)はインタフェース３５０の構成例、（Ｂ）はインタフェース３５０の中で使われる画像構造体ＰＴＮ_ＭＣＢの構成例、（Ｃ）はインタフェース３５０の中で使われるウィンドウ構造体ＰＴＮ_ＷＣＢ の構成例である。インタフェース３５０は、入力画像構造体ImgSrc501，入力ウィンドウ構造体WinSrc502，出力画像構造体ImgDst503 ，出力ウィンドウ構造体WinDst504，フィルタカーネルの横幅KerH505，フィルタカーネルの縦幅KerW506 からなる。画像構造体ＰＴＮ_ＭＣＢ は、画像に関するパラメータを一つの構造体にしたもので、画像の先頭アドレスAddress511，画像の論理的横幅Width512，

画像の論理的縦幅Height513，画像の物理的実体の横幅ａ_Width514，画像の物理的実体の縦幅ａ_Height515，一画素を表すのに用いられるビット数depth516からなる。ここで、論理的幅といっているのは、実際に画素のデータが入っている幅で、物理的実体の幅というのは、アクセスに便利なように論理的幅より大きい１６の倍数にして実際にメモリ上に割り当てる幅のことである。ウィンドウ構造体ＰＴＮ_ＷＣＢ は、画像上におけるウィンドウの先頭ｘ座ｘ_coord521，画像上におけるウィンドウの先頭ｙ座標ｙ_coord522，ウィンドウの横幅Width523，ウィンドウの縦幅Height524 からなる。このように、インタフェース３５０は画像のパラメータなどからなっており、一つの画像を複数のスレーブに分割した場合、画像の大きさは変わるが、パラメータの構造自体は同じにすることができる。

【００１５】図６は、マスタスレーブ間のインタフェース３６０の例である。マスタスレーブ間インタフェース３６０は、マスタ通信３１０とスレーブ通信３２０を結ぶものである。下側にむかうにつれて処理が進む形で記述してある。マスタスレーブ間インタフェース３６０は、ローカルメモリ（例えば１１０）上にデータを置くことでやりとりを実現する。ローカルメモリ上のデータには、ローカルメモリのアクセス権６３１～６３５，画像データ６４１～６４７からなる。ローカルメモリのアクセス権は１ビットで、０がマスタにアクセス権あり，１がスレーブにアクセス権ありを示す。マスタ通信３１０はまず、画像をスレーブの数に分割する画像分割６１１を行い、その後にローカルメモリ上に領域を設けるメモリ確保６１２を行う。このとき、アクセス権６３１と画像データ６４１の領域が設けられ、アクセス権は０に初期化され、すなわちマスタにアクセス権ありがデフォルトとなる。次に最初の１ラインの画像データを送る画像データ送信６１３が行われ、画像データ６４２にデータが送られる。データが送り終わるとアクセス権を１にする処理６１４が行われ、アクセス権６３２は１の値となる。マスタ通信３１０はスレーブ起動６１５を実行し、スレーブを立ち上げる。６１２から615の処理はスレーブの数だけ行う必要があり、ループ判定６１６にて、スレーブ数ｎの回数だけ繰り返す。スレーブ上では、スレーブ通信処理３２０を介し、画像処理３３０が実行開始する。その後、マスタ通信３１０はアクセス権が０にされるのを待つ処理６１７によって、待つ。画像処理３３０は、画像データ６４３を読み出し、演算６６１を行って、結果を画像データ６４３に書き込む。一行分が終わったら、最終行か否かの判定６６２を行い、最終行でない場合は、スレーブ通信処理３２０がアクセス権をマスタに返す処理６５１を行い、マスタがアクセス権をスレーブに渡してくるまで待つ処理６５２に入る。アクセス権が０にされるのを待つ処理６１７を行っていたマスタ通信３１０はアクセス権０を検出すると画像データ６４４を読み込む画像データ受信６１８を行い、１行分の結果を集めたあと、画像データ送信６１９により次の１行分のデータ６４５を送る。１行分送り終わったら、アクセス権６３４を１にする処理６２０を行い、これをスレーブ数ずつ繰り返す処理６２１とさらに画像の縦幅分繰り返す処理６２２により、処理を継続する。マスタがアクセス権をスレーブに渡してくるまで待つ処理６５２に入っていたスレーブ通信３２は、アクセス権６３４がマスタ通信３１０によって１にされると、画像処理３３０に次の行に関する演算６６１を行わさせる。こうして画像全体の処理を進め、最後の行において、最終行か否かの判定６６２で最後の行であると判定されると、スレーブ通信３２０はアクセス権635を０にする処理６５３を行って処理を終了する。マスタ通信３１０もこのときアクセス権６３５が０かどうかを判定する処理６２３で待っているはずで、アクセス権６３５が０であると判定すると最後の行の演算結果である画像データ６４６を集める画像データ受信６２４を行い、これをスレーブの数だけ繰り返す処理６２５で繰り返した後、処理を終了する。このように、マスタ通信３１０とスレーブ通信３２０は、インタフェース３６０を介して、アクセス権６３１～６３５で一行ずつ同期をとりながら画像データ６４１～６４６をやりとりして処理を進めていく。

【００１６】図７は、画面を縦に分割する場合の入力画像の構成例を示したものである。マスタ通信３１０で画像をスレーブの数SlaveNumに分割する画像分割６１１での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、入力画像をスレーブの数に分けるときには、境界部分でカーネルの大きさ分だけ重なり合わせた形で分けなければならない。スレーブの入力画像Slv->ImgSrcは、マスタの入力画像Mst->ImgSrcのパラメータを用いて次のような式で表される。

【００１７】Slv->ImgSrc.Width={Mst->WinSrc.Width-(Mst->KerW-1)}/SlaveNum+(Mst->KerW-1)
Slv->ImgSrc.Height=Mst->WinSrc.Height
Slv->WinSrc.x_coord=0, Slv->WinSrc.y_coord=0
Slv->WinSrc.Width={Mst->WinSrc.Width-(Mst->KerW-1)}/SlaveNum+(Mst->KerW-1)

Ｓｌｖ－＞ＷｉｎＳｒｃ．Ｈｅｉｇｈｔ＝Ｍｓｔ－＞ＷｉｎＳｒｃ．Ｈｅｉｇｈｔ

図８は、画面を縦に分割する場合の出力画像の構成例を示したものである。マスタ通信３１０で画像をスレーブの数ＳｌａｖｅＮｕｍに分割する画像分割６１１での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、出力画像は周囲部分でカーネルの大きさの半分の分だけ計算できない領域が発生する。図ではこの領域をハッチングして表している。スレーブの出力画像Slv->ImgDstは、マスタの入力画像Mst->ImgDstのパラメータを用い

て次のような式で表される。

【００１８】Slv->ImgDst.Width={Mst->WinDst.Width-(Mst->KerW-1)}/SlaveNum+(Mst->KerW-1)
Slv->ImgDst.Height=Mst->WinDst.Height
Slv->WinDst.x_coord=0, Slv->WinDst.y_coord=0
Slv->WinDst.Width={Mst->WinDst.Width-(Mst->KerW-1)}/SlaveNum
Slv->WinDst.Height=Mst->WinDst.Height

図９は、画面を横に分割する場合の入力画像の構成例を示したものである。マスタ通信３１０で画像をスレーブの数SlaveNumに分割する画像分割６１１での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、入力画像をスレーブの数に分けるときには、境界部分でカーネルの大きさ分だけ重なり合わせた形で分けなければならない。スレーブの入力画像Slv->ImgSrcは、マスタの入力画像Mst->ImgSrcのパラメータを用いて次のような式で表される。

【００１９】Slv->ImgSrc.Height={Mst->WinSrc.Height-(Mst->KerH-1)}/SlaveNum+(Mst->KerH-1)
Slv->ImgSrc.=Width->WinSrc.Width
Slv->WinSrc.x_coord=0, Slv->WinSrc.y_coord=0
Slv->WinSrc.Height={Mst->WinSrc.Height-(Mst->KerH-1)}/SlaveNum+(Mst->KerH-1)
Ｓｌｖ－＞ＷｉｎＳｒｃ．Ｗｉｄｔｈ＝Ｍｓｔ－＞ＷｉｎＳｒｃ．Ｗｉｄｔｈ

図１０は、画面を横に分割する場合の出力画像の構成例を示したものである。マスタ通信３１０で画像をスレーブの数ＳｌａｖｅＮｕｍに分割する画像分割６１１での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、出力画像は周囲部分でカーネルの大きさの半分の分だけ計算できない領域が発生する。図ではこの領域をハッチングして表している。スレーブの出力画像Slv->ImgDstは、マスタの入力画像Mst->ImgDstのパラメータを用いて次のような式で表される。

【００２０】Slv->ImgDst.Height={Mst->WinDst.Height-(Mst->KerH-1)}/SlaveNum+(Mst->KerH-1)
Slv->ImgDst.=Width->WinDst.Width
Slv->WinDst.x_coord=0, Slv->WinDst.y_coord=0
Slv->WinDst.Height={Mst->WinDst.Height-(Mst->KerH-1)}/SlaveNum
Slv->WinDst.Width=Mst->WinDst.Width

図１１は、第２のソフトウェア構成例である。アプリケーション３００，マスタ管理１１００，マスタ通信３１０，スレーブ通信３２０，画像処理３３０からなる。アプリケーション３００，マスタ管理１１００，マスタ通信３１０，画像処理３３０は図１におけるマスタプロセッサ１００上で動作するプログラムであり、スレーブ通信３２０，画像処理３３０は、図１におけるスレーブプロセッサ、たとえばスレーブプロセッサ１１０上で動作するプログラムである。これは、残りのスレーブプロセッサ１２０，１３０，１４０でも全く同じ構成となる。アプリケーション３００は、インタフェース１１１０を用いてマスタ管理１１００を呼び出し、データをやり取りする。マスタ管理１１００は、インタフェース３５０を用いてマスタ通信３１０または画像処理３３０のどちらかを呼び出し、データをやり取りする。マスタ通信３１０はマスタスレーブインタフェース360を用いてスレーブ通信３２０を呼び出し、データをやり取りする。スレーブ通信３２０は、アプリケーション３００がマスタ通信３１０とデータをやり取りするインタフェースと全く同一のインタフェース３５０を用いて画像処理３３０を呼び出し、データをやり取りする。このように、マスタ管理１１００からマスタ通信３１０または画像処理３３０を呼び出すインタフェース３５０と、スレーブ通信３２０から画像処理３３０を呼び出すインタフェース３５０を同一にしておくことにより、画像処理３３０をマスタで実行させることも、スレーブで実行させることも可能となる。

【００２１】

【発明の効果】本発明によれば、並列プロセッサで動作するプログラムを、単一プロセッサで実行するように、再コンパイルなどを行うことなく、最小の工数で実現でき、並列プロセッサの装置を、単一プロセッサの装置に変更できるため、装置の小型化／装置コストの削減という効果がある。

【図面の簡単な説明】

【図１】本発明からなる画像処理システムの一実施例を示す図である。

【図２】本発明からなる画像処理システムの他の実施例を示す図である。

【図３】本発明からなる画像処理システムのソフトウェア部に関する一実施例を示す図である。

【図４】本発明からなる画像処理システムのデータ通信を説明する図である。

【図５】本発明からなる画像処理システムのインタフェースの一実施例を示す図である。

【図６】本発明のインタフェースのフローチャートを示す図である。

【図７】本発明の画面を縦に分割する場合の入力画像の一構成例を示す図である。

【図８】本発明の画面を縦に分割する場合の出力画像の一構成例を示す図である。

【図９】本発明の画面を横に分割する場合の入力画像の一構成例を示す図である。

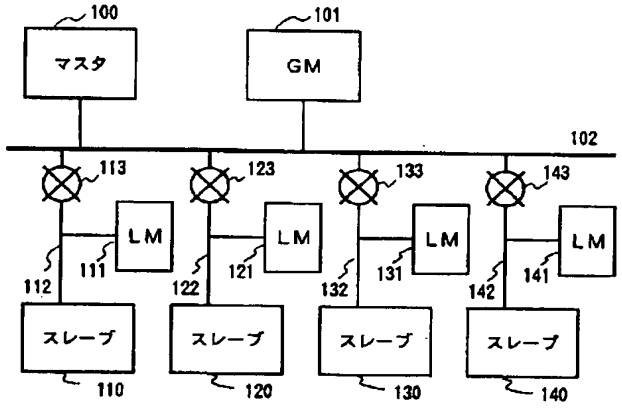
【図１０】本発明の画面を横に分割する場合の出力画像の一構成例を示す図である。

【図１１】本発明からなる画像処理システムのソフトウェア部に関する他の実施例を示す図である。

【符号の説明】
１００…マスタプロセッサ、１０１…グローバルメモリ、１０２…マスタバス、１１０，１２０，１３０，１４０…スレーブプロセッサ、１１１，１２１，１３１，１４１…ローカルメモリ、１１２，１２２，１３２，１４２…スレーブバス、１１３，１２３，１３３，１４３…バススイッチ、３００…アプリケーション、３１０…マスタ通信、３２０…スレーブ通信、３３０…画像処理、３５０…インタフェース、３６０…マスタスレーブインタフェース、５０１…入力画像構造体、５０２…入力ウィンドウ構造体、５０３…出力画像構造体、５０４…出力ウィンドウ構造体、５０５…フィルタカーネルの横幅、５０６…フィルタカーネルの縦幅、５１１…先頭アドレス、５１２…論理値横幅、５１３…論理値縦幅、５１４…物理的実体の横幅、５１５…物理的実体の縦幅、５１６…ビット数、５２１…ウィンドウの先頭Ｘ座＊標、５２２…ウィンドウの先頭Ｙ座標、５２３…ウィンドウの横幅、５２４…ウィンドウの縦幅、６１１…画像分割、６１２…メモリ確保、６１３，６１９…画像データ送信、６１４，６２０…アクセス権を１にする処理、６１５…スレーブ起動、６１６…ループ判定、６１７…アクセス権が０にされるのを待つ処理、６１８，６２４…画像データ受信、６２１，６２５…繰り返す処理、６２２…縦幅分繰り返す処理、６２３…アクセス権が０かどうか判定する処理、６３１，６３２，６３３，６３４，６３５…アクセス権、641，６４２，６４３，６４４，６４６…画像データ、６４５…１行分のデータ、６５１…アクセス権をマスタに返す処理、６５２…アクセス権をスレーブに渡してくれるまで待つ処理、６５３…アクセス権を０にする処理、６６１…演算、６６２…最終行か否かの判定、１１００…マスタ管理、１１１０…インタフェース。

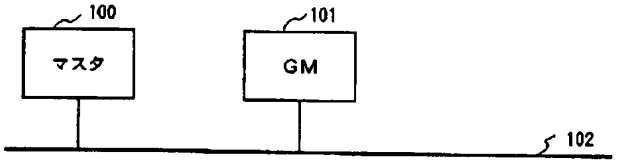
【図１】

図 1



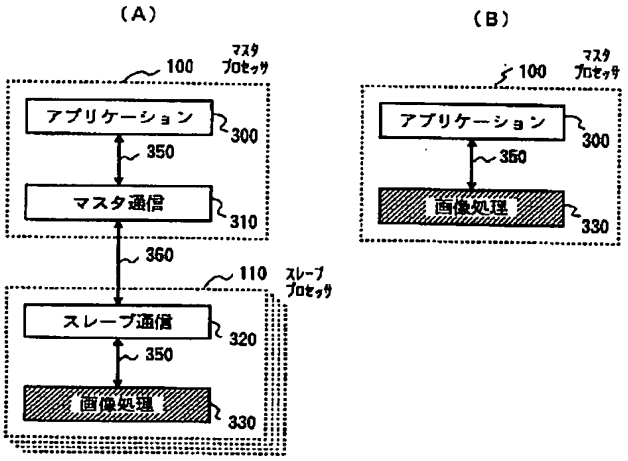
【図２】

図 2



【図３】

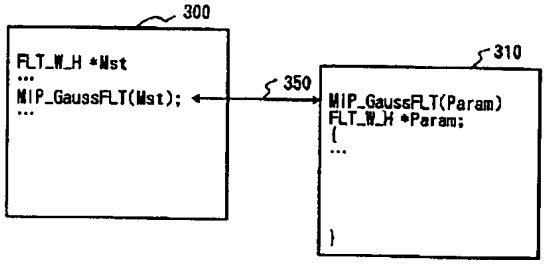
図 3



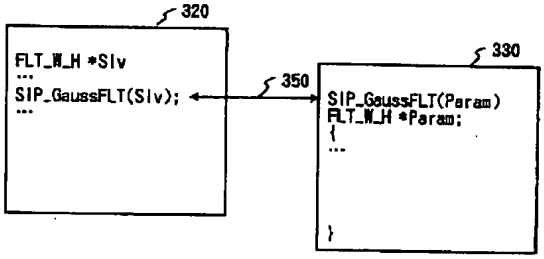
【図４】

図 4

(A)



(B)



【図５】

図 5

(A)

```
typedef struct{
        PTN_WCB         ImgSrc;     ～501    /* 入力画像 */
        PTN_WCB         WinSrc;     ～502    /* 入力ウィンドウ */
        PTN_WCB         ImgDst;     ～503    /* 出力画像 */
        PTN_WCB         WinDst;     ～504    /* 出力ウィンドウ */
        unsigned short  KerH;       ～505    /* カーネル横幅 */
        unsigned short  KerW;       ～506    /* カーネル縦幅 */
} FLT_W_H;
```

(B)

```
typedef struct{
        unsigned long   Address;    ～511    /* 先頭アドレス */
        unsigned long   Width;      ～512    /* 論理的横幅 */
        unsigned long   Height;     ～513    /* 論理的縦幅 */
        unsigned long   a_Width;    ～514    /* 物理的実体の横幅 */
        unsigned long   a_Height;   ～515    /* 物理的実体の縦幅 */
        char            depth;      ～516    /* 画素値の深さ */
} PTN_WCB;
```

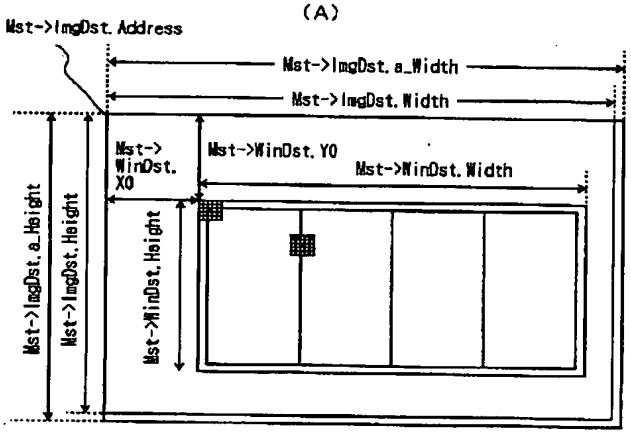
(C)

```
typedef struct{
        short           x_coord;    ～521    /* 先頭のｘ座標 */
        short           y_coord;    ～522    /* 先頭のｙ座標 */
        unsigned short  Width;      ～523    /* ウィンドウの横幅 */
        unsigned short  Height;     ～524    /* ウィンドウの縦幅 */
} PTN_WCB;
```

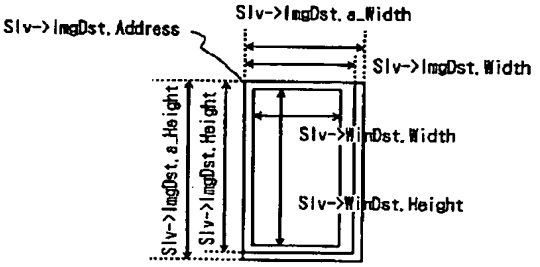
【図８】

図 8

(A)



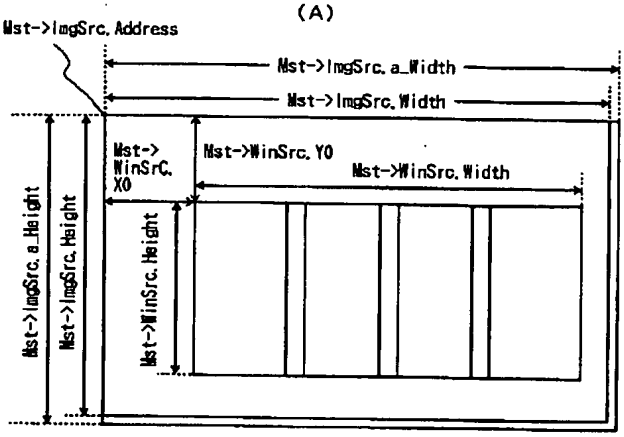
(B)



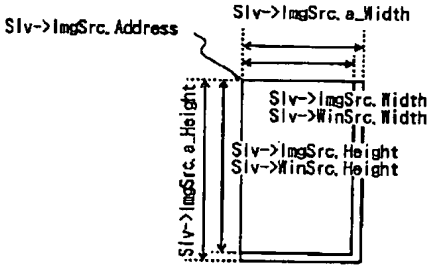
【図７】

図 7

(A)



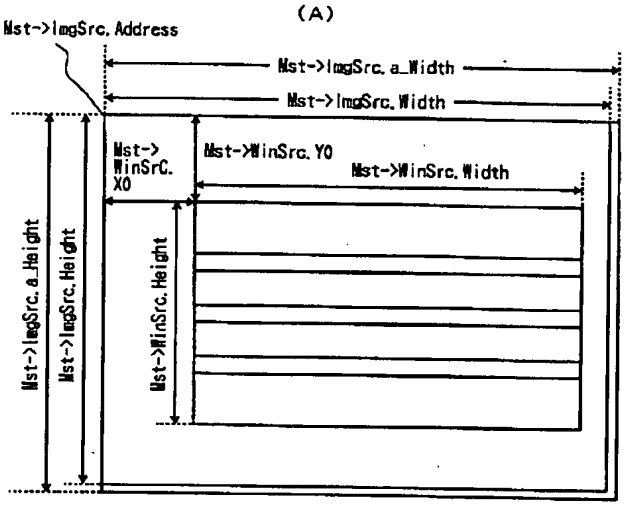
(B)



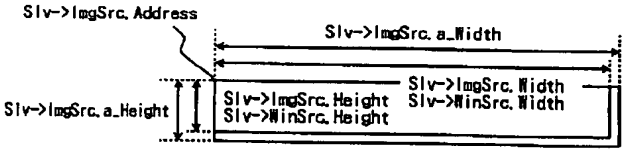
【図９】

図 9

(A)

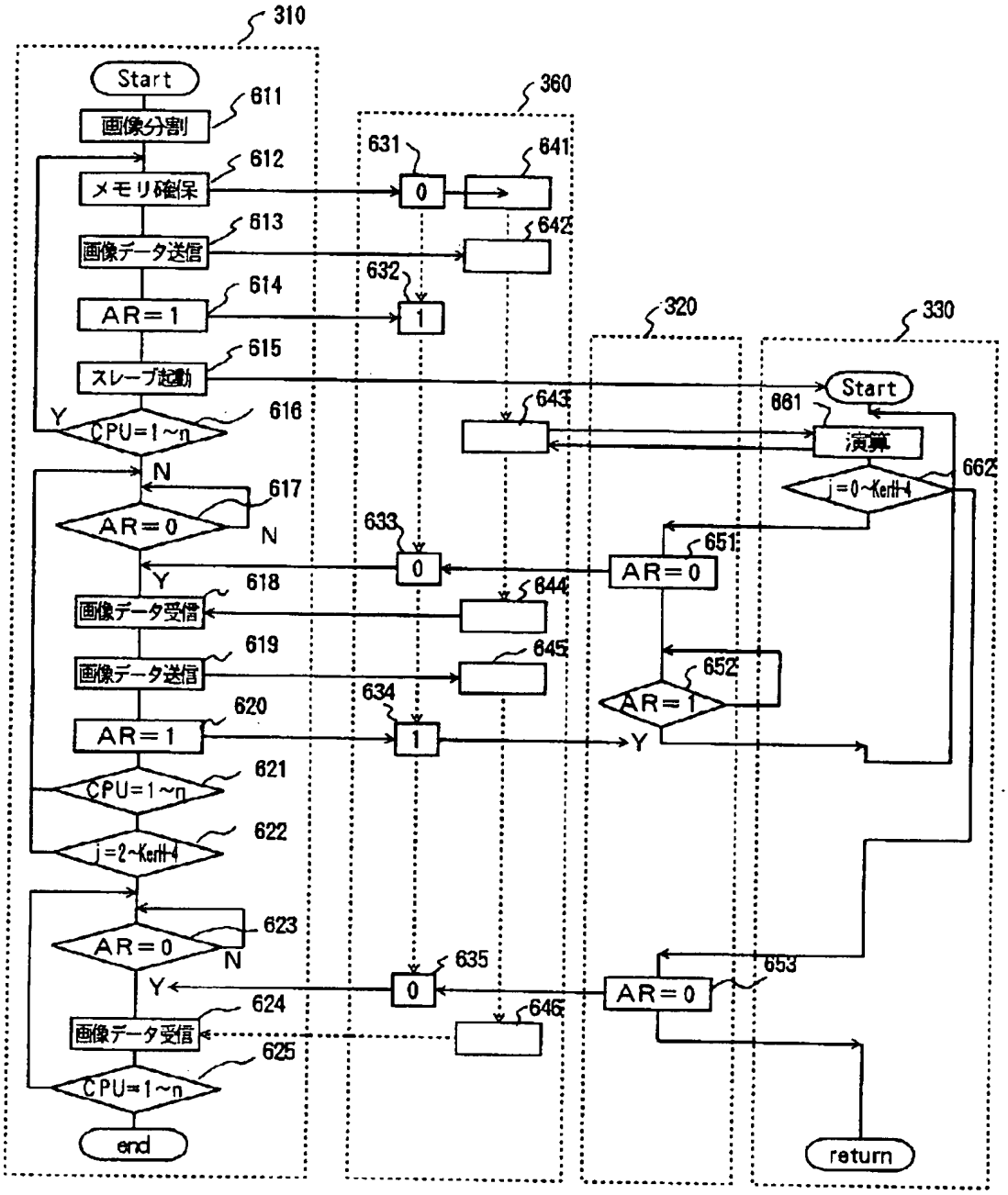


(B)



【図６】

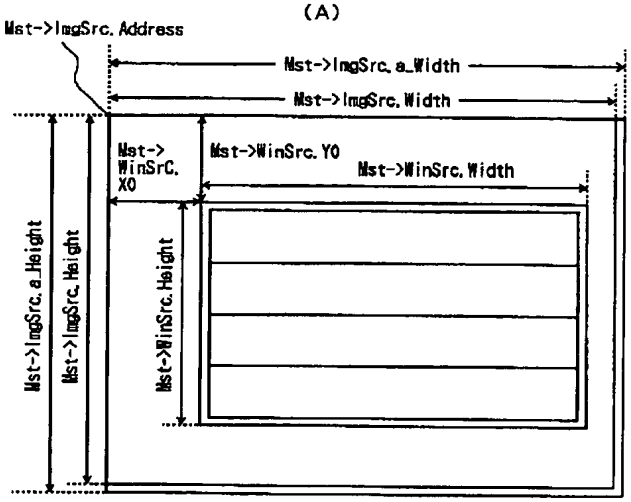
図 6


310
Start
611
画像分割
612
メモリ確保
613
画像データ送信
614
AR＝1
615
スレーブ起動
616
CPU=1～n
Y
N
617
AR＝0
N
Y
618
画像データ受信
619
画像データ送信
620
AR＝1
621
CPU=1～n
622
j=2～Kerｎ-1
623
AR＝0
N
Y
624
画像データ受信
625
CPU=1～n
end
360
631
0
641
632
1
642
633
0
643
634
1
644
635
0
645
646
320
651
AR＝0
652
AR＝1
Y
653
AR＝0
330
Start
661
演算
662
j=0～Kerｎ-1
return

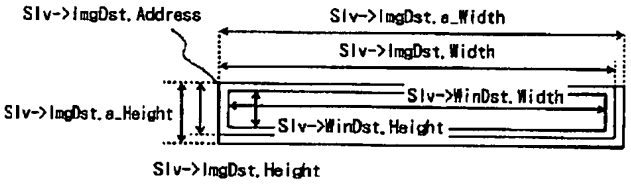
【図１０】

図 10

(A)

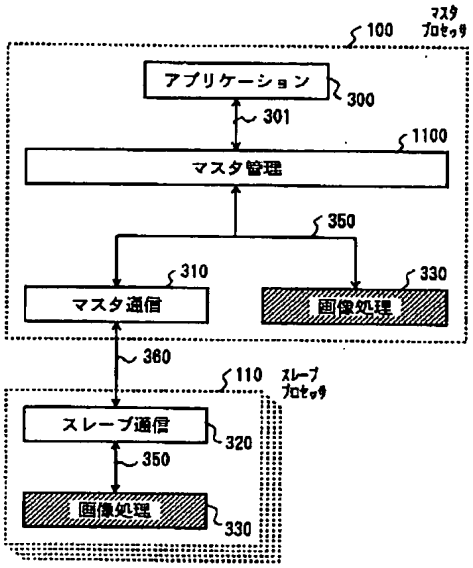


(B)



【図１１】

図 11



フロントページの続き

(72)発明者　高根　淳
茨城県ひたちなか市大字市毛882番地　株式会社日立製作所計測器事業部内

(72)発明者　佐藤　典夫
茨城県ひたちなか市大字市毛882番地　株式会社日立製作所計測器事業部内

Ｆターム(参考)　5B045 AA01 BB31 GG11
5B057 CH04 CH14 CH16